❖ Features / Advantages / What is ASP.NET :-

- Asp.NET is an upgraded version of classical ASP (Active Server Pages).
- It is a server side technology and a very important part of Microsoft's .NET framework.
- It provides services that allow the creation, deployment & execution of web application on the server.
- It can be used to build powerful web application using the Common Language Runtime (CLR).
- ASP.NET runs inside the IIS (Internet Information Server) which is the Microsoft's internet server freely available with windows.
- ASP.NET comes with built-in web forms controls which are responsible for generating the user interface and are same as the HTML controls.
- An ASP.NET application will have all code files, pages, handles, modules and executable code that can be invoked or run in a given virtual directory on web server.
- Each ASP.NET application is executed within a .NET framework domain which guaranties class isolation, security sendboxing and static variable isolation.

❖ Advantages of ASP.NET :-

1. Separation of code from HTML :

   It is easier for team of programmers & designers to collaborate efficiently.
2. Support for compiled languages :

   It uses all compiled languages & the pages are pre-compiled into byte code & JIT (Just-In-Time) when first requested, then after they are fully compiled & cached till the source changes.
3. Use services provided by the .NET framework.
4. Graphical development environment :

   It uses/provides visual studio .NET for drag & drop of control & to set properties.
5. State Management :

   It provides different state management techniques like session, cookies, view state, application state.

6. Update files while the server is running :

   Components of your application can be updated while the server is online & clients are connected.

7. XML (Extensible Markup Language) based configuration files.

❖ Client Side Scripting & Server Side Scripting :

Client Side Script:-

1. In client side scripting, the scripts are executed on client machine using browsers.
2. Because it execute locally, code is visible in HTML page.
3. Because the code is visible, it is less secure.
4. It is light in weight & executes faster.
5. It is usually used for client side validation for HTML form.
6. Language like JavaScript & VbScript is client side scripts.

Server Side Script:-

1. Here the scripts are executed on the server & the o/p is generated in HTML form & then sends to the client browser.
2. Here the code is not visible.
3. Here everything is on the server so it is very secure.
4. It takes more time & load to execute.
5. It is useful for server side validation involving communication with database.
6. PHP, ASP are server side scripting.

❖ ASP.NET webpage coding model :-

- ASP.NET provides 2 types of coding model as follows –
  1. Inline Code (Single File)
  2. Code Behind File

  Both the models are functionally same at runtime. There is no difference between them in performance.

1. Inline Code:-

- Here we have the ASP.NET code in a single file .aspx along with the html code.
- The ASP.NET code is written in a script tag along with the attribute runat =”server”.

```
<script runat=”server”>
        ASP.NET code
</script>
<body>
 .
 .
 .
</body>
```

- Bydefault when we create a new aspx page, it is created using inline code file – default.aspx .
- Example:

  Default.aspx

```
<@page Language=”VB”>
<script runat=”server”>
        Protected sub button1_click (ByVal sender as object, ByVal e as system.Eventargs)
                Msgbox (“welcome”)
        End sub
</script>
<html>
        <head runat=”server”>
```

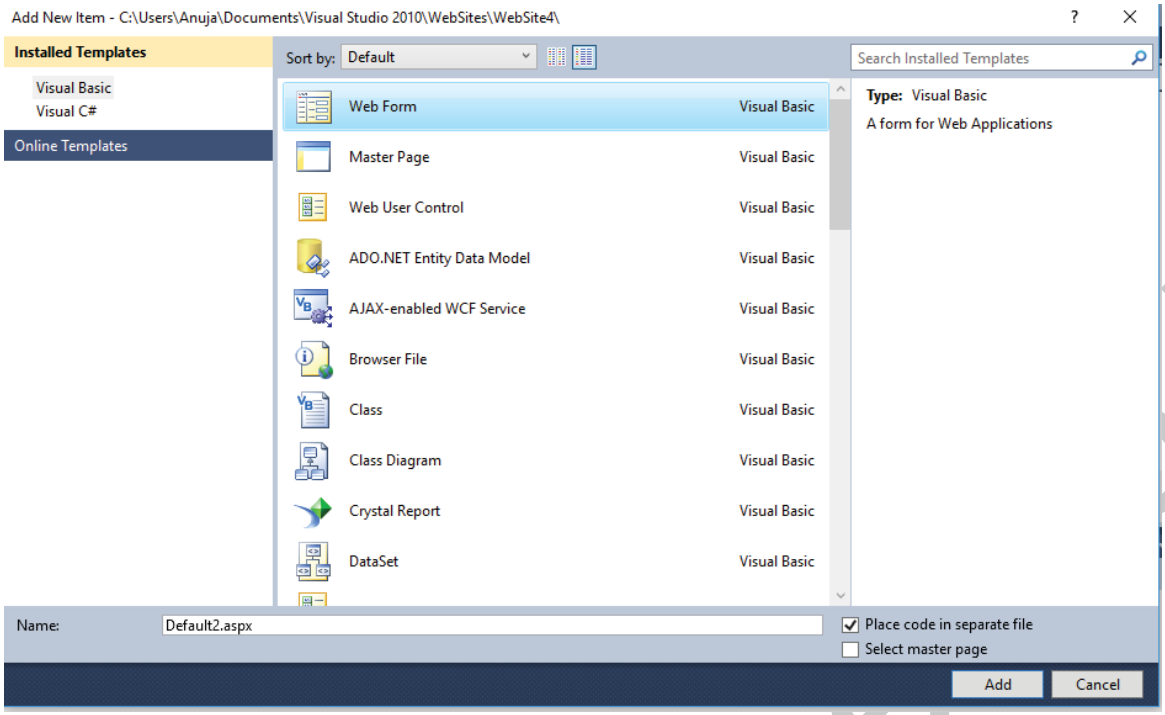
```
        <title>Default page </title>
    </head>
    <body>
        <form runat="server">
            <asp:Button runat="server" ID="Button1"
            value="Button" />
        </form>
    </body>
</html>
```

- Advantages of inline file :-
    1. It is convenient to keep the code & markup in same file if the code is very less.
    2. Pages written using inline file are easier to deploy or send to another developer because there is only one file.
    3. The total files in the project will be less.

2. Code behind File:-

- The code behind model allows us to keep the html code in one file and the programming code in another file.
- Both these files are then linked together to run the web application.
- The extension of another file (code file) depends on the programming language that we select.
- Example: if we select vb, file will be default.aspx.vb and if we select c#, file will be default.aspx.cs
- For code behind file we need to keep in mind following points –

    1. To create a new page we go to add new item in your solution explorer.
    2. It opens a dialog at new item.
    3. We select web form from that dialog box. Bydefault it will be named as default.aspx and below it we will select the language.
    4. Now to follow code behind model we select the following checkbox.



- Example:- default.aspx

```
<@page Language=”VB” CodeBehindFile=”Default.aspx.vb” Inherits=”Default”>
<! Doctype HTML>
<html>
    <head runat=”server”>
        <title>Default Page </title>
    </head>
    <body>
        <form runat=”server”>
            <asp:Label ID=”lblname” runat=”server” >
                Name :
            </asp:label>
            <asp:Textbox ID=”txtname” runat=”server”>
            </asp:Textbox>
            <asp:Button ID=”btnsubmit” runat=”server” text=”submit” />
        </form>

    </body>

</html>
```

- Default.aspx.vb :-

```
Imports System.Web
Partial class Default Inherits System.Web.UI.Page
    Protected sub btnsubmit_click(ByVal sender As object, ByVal c As
    System.EventArgs)
                Msgbox(txtname.text)
    End sub
End class
```

- The code file attribute specifies the name of the separate class file.
- Inherits attribute specifies the name of the class in the code behind file.

- Advantages of code behind model:-

1. Code behind page offers a clean separation markup and code.
2. Code can be reused for multiple pages.
3. We can have a designer working on a markup while the programmer writes the code.

❖ Why we use partial class from ASP.NET 2.0 ?

- In earlier version of ASP.NET a single class was in single file i.e a class can not be split across multiple files.
- From ASP.NET 2.0 we have a new feature called partial class which allows us to allocated different developers to develop the code for different functionalities that are available for single class.
- These functionalities can be developed in partial classes and then compiled to from required assembly.
- The partial class is inherited from a base page class (System.Web.UI.Page).
- The .aspx file contains and inherits attribute that points to the code behind partial class.
- When the page is compiled ASP.NET creates a new partial class for .aspx file.
- Finally both the generated partial classes are compiled into a single class at runat.

❖ Types of files in ASP.NET :-

1) .asax :-

- It is a file that contains the code which is applied globally across the application.
- Eg. global.asax (root directory)

2) .ascx :-

- It is a file that defines a customized, reusable user control. (root or sub directory)

3) .ashx :-

- It is a file that contains code for implementing IHttpHandler interface. (root or sub directory)

4) .asmx :-

- It is a file that contains XML web services that are available to other web applications. (root or sub directory)

5) .aspx :-

- It is a ASP.NET webforms file that contains web control, HTML code & other logic. (root or sub directory)

6) .config :-

- It is an XML configuration file that contains XML elements to configure various features of ASP.NET. (root or sub directory)
- Eg. web.config

7) .master :-

- It is a master page file that defines the layout for other web.pages . (root or sub directory)

8) .sitemap :-

- It is an XML file that contains the structure of the website. (root directory)

9) .skin :-

- It is used to determine & display formatting of web application. (App_Themes)

10) .sln :-

- It is a solution file for visual studio web developer project.

❖ ASP.NET page life cycle

In ASP.NET the page life cycle consist of some ordered no. of events which are as follows

1. Page Request :-
   - The page request occurs before the page life cycle begins .
   - When the page is requested by a user,ASP.NET determines whether the page requested needs to be parsed and compiled or a cached version of the page can be send in response to the request.

2. Start :-
   - In this state the page properties such as request and response are set.
   - It also determines whether the request is a postback or a new request and sets the IsPostBack property.

3. Page Initialization :-
   - Here the controls on the page are available and each control's ID property is set.
   - Any themes available are also applied at this stage.

4. Load :-
   - In this stage the page is loaded with the information set bydefault or with the information recovered from application or view state in case of a PostBack.

5. Validation :-
   - Here all the validator controls are called using validate method which sets the IsValid property for a control.

6. PostBack Event Handling :-
   - If the page is PostBack then relevant event handlers responsible for PostBack are called.

7. Rendering :-

- In this stage, the page calls the render method for every control and writes the output stream of the page response property. (in the form of pure HTML)

8. Unload :-

- Unload is called after the pages successfully rendered, sent to the client and is ready to be discarded.
- Page properties are unloaded and required cleanup are performed.

❖ Sequence of events in page life cycle :-

1. PreInit :-
   - This event is used for the following purposes –
     1. To check whether the page is processed for first time or not using IsPostBack .
     2. Create or recreate dynamic controls.
     3. Set themes or master pages dynamically.

2. Init :-
   - This event is used to initialize all the controls and their properties.

3. InitComplete :-
   - This event is raised by the page object.
   - All the task that require complete initialization are processed here.

4. PreLoad :-
   - We can use this event to perform processing that is required before the Load event.

5. Load :-
   - It calls OnLoad event method of the page and then recursively for all the controls and child controls.
   - This event is usually used for establishing database connection.

6. Control Events :-
   - It is raised for specific control such as button's click event, textbox's textchange event etc.

7. Load Complete :-
   - This event is used for task that require complete loading of all the controls on the page.

8. PreRender :-

- It occurs for every control on the page.
- This event is used for making final changes to the contents of the pages for controls.

9. SaveStateComplete :-

- This event is used for saving view state for required controls without making any changes to controls.

10. Render

11. UnLoad :-

- In this event all this controls and then the page itself will be cleaned up and any open database connections or files will be closed.

❖ Directory structure in ASP.NET web application

In ASP.NET we can save the files in any folder structure within the application root directory.

But to make the work easier, ASP.NET reserves certain files and folders name that can be used for specific types of contents.

We can add reserves folder by right clicking on solution explorer and selecting “Add ASP.NET Folder”.

The following are reserved folders in ASP.NET :

1. App_Browsers :-
   - It contains files which have definations for particular site.
2. App_code :-
   - It contains source code for utility classes and business objects that are a part of our application.
   - It will be used for data access abstraction code, model code and business code.

3. App_Data :-
   - It contains database files such as access’s .accdb file, .mdb file and sql’s .mdf file.

4.App_Themes :-
   - It holds files for different themes to be applied to the website.
   - Usually it contains .skin and .css file.

5. App_GlobalResources :-
   - It contains resources that are associated with specific page or specific control.
   - Eg. for a file order.aspx and its resource file .resx will contain particular version for particular country.

6. App_WebReference :-
   - It holds the discovery files and WSDL files for references to webservices used within application.

7. Bin :-

- It contains compiled code (.dll files) for controls, components, etc that we want to refer in our application.
- Any classes represented by the Bin folder are automatically referenced in our application.

❖ Page Directives :-

- While creating a webpage in ASP.NET we can declaratively set a number of attributes about the page.
- Following are the directives are available on a .aspx page.

1. @page :-

- This directive is used to assign page specific attributes that are used by the web forms page parser and compiler to inform them how the page will be created.
- There can be only one @page directive in single file.
- It can be placed anywhere in the file but it is preferable to keep it at the top.

Attributes of @Page directives :-

1. Language :- Specifies the name of .net language used to compile source code.

2. Inherits :- Specifies the name of code behind class.

3. CodeBehindFile :- Specifies the name of file containing the code behind class.It has an extension .aspx.vb or .aspx.cs .

4. ClassName :- The name of the class the page is derived from.

5.EnableViewState :- A Boolean property which specify whether the view state is maintains or not.

6. EnableSessionState :- A Boolean property which specify whether the page can have access to session object or not.

7. ErrorPage :- It specifies valid URL for an error page, the page is redirected when an error occurs.

8. AutoEventWireUp :- Specifies a Boolean property whether the page events are handled automatically or not. If not then critical events like Page_Load must be explicitly handled by developer.

Eg: <%@page Language="VB" CodeBehindFile="Default.aspx.vb" Inherits="Default" EnableViewState="true" AutoEventWireUp="false" >

2. @Import :-

- This directive is used to explicitly import a namespace on a page.
- This will make all the classes and interfaces contained in the namespace available to the code on page.

Eg: <%@Import namespace=”value” >

- Following are the namespace.

A) System

B) System.Data

C) System.Configuration

D) System.Web

E) System.Web.UI

F) System.Collections

- We can import only one namespace using @Import.
- To import multiple namespace use multiple@Import directive.

3. @Implements :-

- It is used to implement a .net interface on a page.
- By implementing an interface, the page will support defined properties, methods, events of that interface.
  E.g. <%@ Implements Interface="interface_name" %>

4. @Assembly :-

- It is used to reference an assembly directly so that classes and interfaces become available to the page.

Eg. <%@Assembly Name=”assemblyname”>

Or

<%@Assembly src=“path”>

- This directive is usually not used because any assembly available in ASP.NET application's bin directory is directly available to the application and compiled with the page.

5. @Register :-

- When we are adding a customized server control, we need to tell the compiler something about that control.
- If the compiler does not know what namespace the control contains then it will not be able to recognize the control and will give an error.
- Thus to inform the compiler, the information it needs, we use @Register directives.
- There are 2 forms of @Register directive depending on the location of the customized control.

<%@Register TagPrefix="prefix" TagName="name" Src="Source" %>

<%@Register TagPrefix="prefix" TagName="name" Assembly="assembly_name" NameSpace="namespace" %>

Eg 1.

<%@Register TagPrefix="Ecomm" TagName="Header" Src="usercontrols\Ecomm.ascx" %>

- Now after registering this control we can use the control using the following syntax:
  <Ecomm:Header ID="head1" runat="server">

6. @Control :-

- The @Control directive is used to assign controls specific attribute that are used by the webforms page parser and compiler to show how the user control is created.

Attributes :-

1. EnableViewState :- A Boolean property which specify whether the view state is maintained or not.

Eg . <%@Control EnableViewState=”true” Language=”VB” Src=”SourcePath” AutoEventWireUp=”true” Inherits=”ClassName” %>

7. @Application :-

- This directive is define in global.asax file and also supports the following attribute :-

1. Inherits :- It allows to name a .net class that will be used by the global.asax as the base class.

2. description :- It adds some descriptive text to give information about the global.asax .

❖ RoundTrip :-

- Most of the webpages required processing on the server.
- Eg :- Consider a Product.aspx page which is used to check the availability of the product. When the user hits the submit button the page is again sent to the server to find the availability of the product. This kind of functionality is achieved by handling server controls.
- Whenever a user interaction requires processing on the server, the webpage is posted back to the server.
- The page is posted back to the server is processed there & returned back to the browser.
- This sequence of processing is called roundtrip (Postback).
- Webpages are recreated with every roundtrip.
- When the server finishes processing and sends the pages to the browser, it discards the page information and frees all the server resource of that request.
- In this way a web application can support thousands of users simultaneously.
- The next time the page is posted, the server recreates it and process it again because webpages are stateless. (Because, HTTP protocol is stateless).
- Sometimes when a user interaction requires processing such that the entire page must not be recreated then in such cases we need to maintain a state of the webpage.
- The user request for the page for the 2nd time then it is call postback.

❖ Global.asax file :- (Application file)

- Global.asax file resides in the root directory of an ASP.NET application & is called the ASP.NET application file.
- It contains the code the code that is executed when events such as start of an application, error in application are raised by the application.
- Events states such as session & application state are specified in this file and are applied to all the resources of the application.
- Eg :- If a state application variable is define in this file then all .aspx files within the root directory can access this variable.
- This file does not contain any HTML or ASP.NET tags. It simply contains the code like the code behind file.
- The code contains the method with predefined names.
- Eg :- Application_start, Application_end, Session_end, Session_start, Application_error.
- This file can not be viewed by external users and can not be downloaded.
- This file is compiled when the application is requested for the first time and it is compiled to a class that extends the HTTP application class.
- If there are any changes then it is detected automatically by asp.net and we do not have to recompile the file.
- There can be only one Global.asax file and it is compulsorily placed in application's root directory.
- Eg:
  frmGlobal.aspx : -

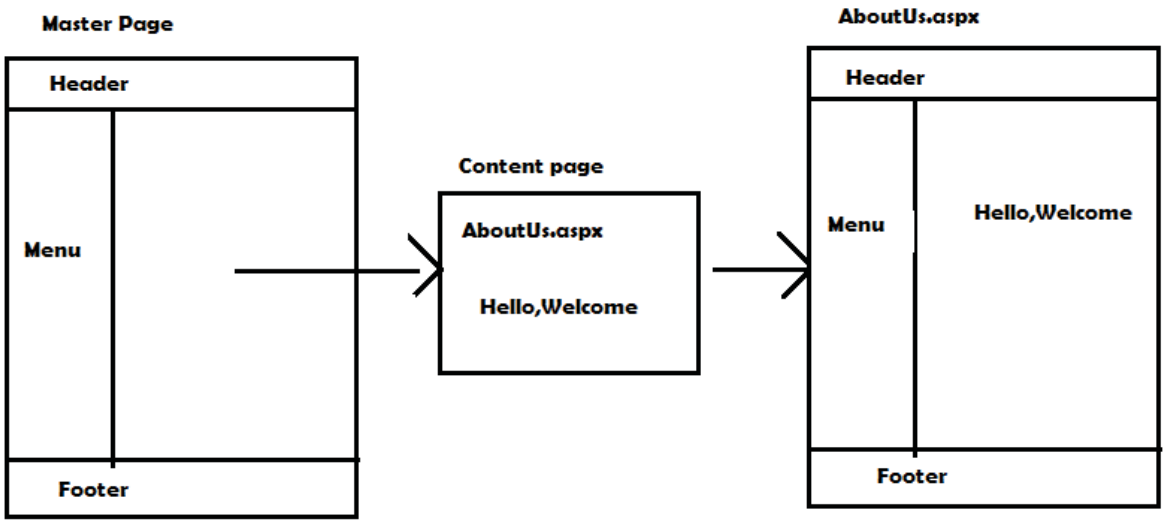
```
Partial Class frmGlobal
        Inherits System.Web.UI.Page

Protected Sub Page_Load (ByVal Sender As object, ByVal e As
System.EventArgs) Handles Me.Load
        Response.Write("Company Name=" & Application
("CompName"))
         Response.Write("<br> counter=" & Application("cnt"))
End Sub
End Class
```

Global.asax : -

```
<Script runat=”server”>
   Sub Application_Start (ByVal Sender As object, ByVal e As
System.EventArgs)
          ‘code that runs on application start_up

          Application (“CompName”) = ”xcel Infotech”
          Application (“cnt”) = 0
   End Sub
   Sub Application_Error(ByVal Sender As object, ByVal e As
System.EventArgs)
          ‘code that runs when an unhandled error occurs
          Response.Redirect (“Error.aspx”)
   End Sub

   Sub Session_Start (ByVal Sender As object, ByVal e As
System.EventArgs)
          ‘code that runs when a session is started
          Application (“cnt”) = Application (“cnt”) + 1
   End Sub



   Sub Session_End (ByVal Sender As object, ByVal e As
System.EventArgs)
     ‘code that runs when a session ends
     Application("cnt")=Application("cnt") -1
    End Sub
</script>
```

❖ Master Pages :-

- Master pages allow us to create a consistent look and behavior for all the pages in our application.
- This feature was introduced with ASP.NET 2.0.
- It provides a template for other pages with shared layout and functionalities.
- A master page can contains markups, controls, banners, navigation and other elements that we want to include on all the pages of our website.
- This makes our website more maintainable and also avoids duplication of code.
- The webpages that inherits the properties that define in master pages are called content pages.
- The content pages display their own property as well as the properties inherited from master pages.
- A master page specifies a ContentPlaceHolder for content which can be overridden by content pages.
- When the user requests the content page, the ASP.NET combines the pages to produce the output having the layout of the master page with content of content page.
- Master pages can be nested. We can also create multiple master pages to define different layouts for different section of our website.
- Eg: There will be different master page for admin side, client side and other users.
- The extension of master page file is .master. It also has a code behind file.
- The master page can not be run directly.



- E.g :- MasterPage1.master

```
<%@Master  %>
<html>
        <head runat=”server”>
                <title>Master Page Demo</title>
        <asp:ContentPlaceHolder ID=”head” runat=”server”>
        </ asp:ContentPlaceHolder>
        </head>
        <body>
                <form runat=”server”>
                        <asp:ContentPlaceHolder ID=”ContentPH1”
        runat=”server”>
                        </ asp:ContentPlaceHolder>
                </form>
        </body>
</html>
```

- Now we can call this master page on the content page default.aspx in the following way.

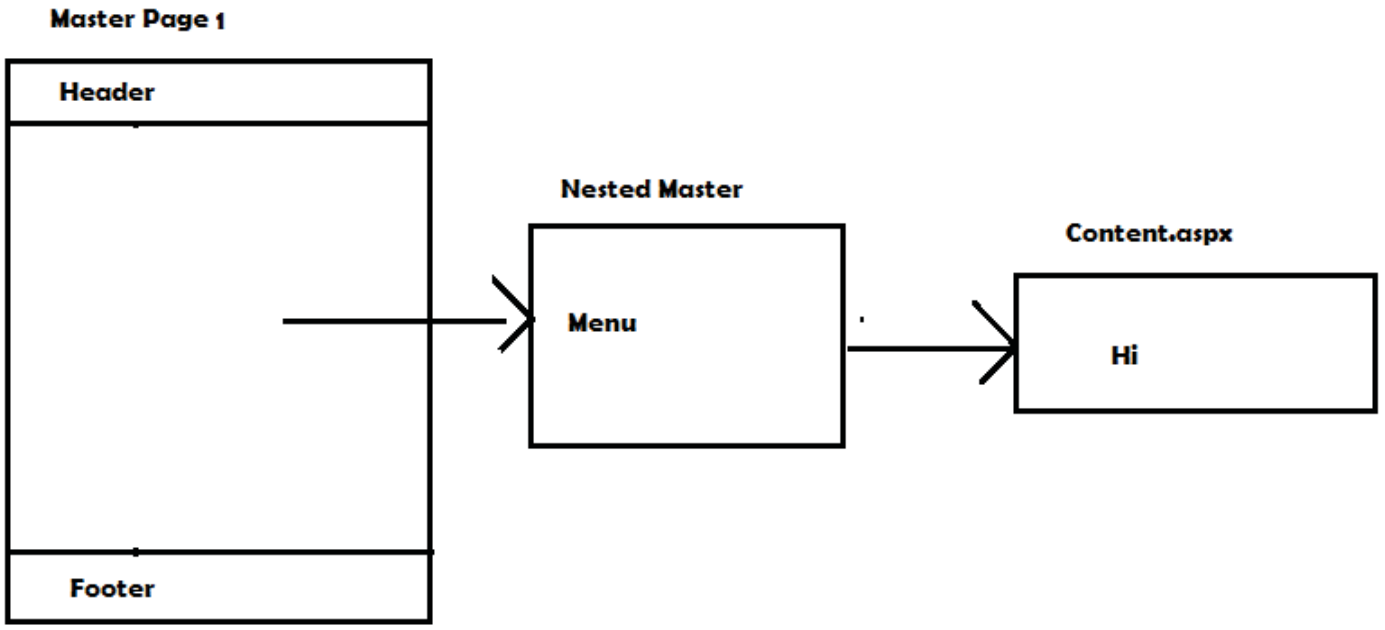
  Default.aspx :-

```
<%@Page MasterPageFile=”MasterPage1.master” Language=”VB”
Inherits=”Defaults” CodeBehindFile=”Default.aspx.vb” %>

    <asp:Content ID=”Content1” runat=”server”
    ContentPlaceHolder=”head”>

    </asp:Content>

    <asp:Content ID=”Content2” runat=”server”
    ContentPlaceHolder=”ContentPH1”>

    </asp:Content>
```

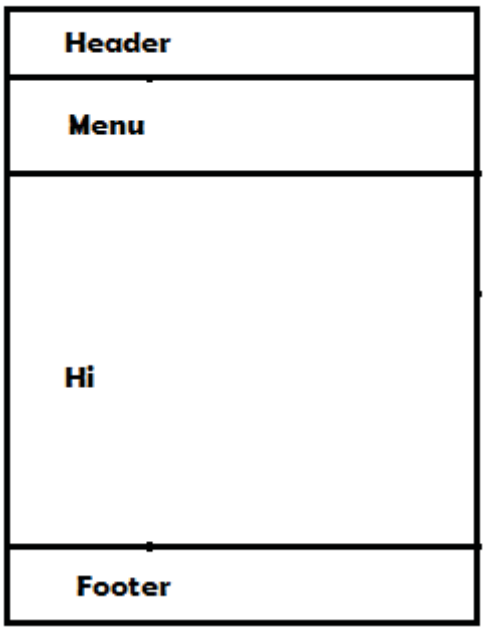
- The content tag’s content will be rendered in <asp:ContentPlaceHolder> whose ID is specified by ContentPlaceHolderID.

❖ Nested Master Page :-

- When one master page references another as its master. It is set to be a nested master page.
- E.g : We can create one master page for the entire website which has only header and footer and then nested the master pages within it for individual sections or users.
- A number of nested master pages can be integrated into a single master.
- There is no limitation on the number of nested master pages that can be created.
- Also the depth of nesting does not impact the performance of the application.
- The advantage of creating child master is that the overall look and feel of the application or website can be made consistent and the child master will give some uniqueness depending on the uses or different section.
- It contains all controls that are mapped to ContentPlaceHolder on parent master page.
- It also has its own ContentPlaceHolder to display the content of child pages.



Nested Master Page



- E.g MasterPage1.master

```
<%@Master Language="VB" %>
<html>
    <head>
        <title>My Nested Page</title>
    </head>
    <body>
        <form>
            <table width="100%" border="2">
                <tr>
                    <td>
                        <img src="images/banner.jpg"
                        width="1024px" />
                    </td>
                </tr>
                <tr>
                    <td>
```

```
                    <asp:ContentPlaceHolder
                ID=”ContentPH1”
                runat=”server”>
                 </asp:ContentPlaceHolder>
                </td>
            </tr>
            <tr>
                <td>
                    Footer
                </td>
            </tr>
        </table>
    </form>
</body>
</html>
```

- Nestedmaster.master :-

```
<%@Master Language=”VB” MasterPageFile=”MasterPage1.master %>
    <asp:Content ID=”NestedContent” runat=”server”
    ContentPlaceHolder=”ContentPH1”>
    <table>
        <tr>
            <td>MenuItem1</td>
            <td>MenuItem2</td>
        </tr>
    </table>
    <table>
        <tr>
            <td>
```

```
                    <asp:ContentPlaceHolder
                    ID="NestedMenu" runat="server">
                    <asp:ContentPlaceHolder>
                </td>

            </tr>

        </table>

    </asp:Content>
```

- Content.aspx :-

```
<%@Page MasterPageFile="Nestedmaster.master"
Language="VB" Inherits="Content" CodeBehindFile="Content.aspx.vb" %>
    <asp:Content ID="Content1" ContentPlaceHolder="NestedMenu"
runat="server">
            <table>
                <tr>
                    <td>
                        Welcome to my nested page
                    </td>
                </tr>
            </table>
    </asp:Content>
```

- Advantages :-

  1. You can give some uniqueness to the web application by creating different child master pages for different regions of the website or different users.

- To find a control on master page :-

```
    Me.Master.FindControl ("ControlName")
```

❖ Themes :-

- Themes are similar to css and enable us to define the visual styles across a webpage. It allows us to apply styles, graphics & other css files on different pages of an application.
- It can be deifne as – "A collection of properties, settings that allows us to define the look of pages and controls and then apply the look consistently across all pages or across the entire web application on a server".
- In ASP.NET, We have a folder with predefine name – App_Themes for creating & storing themes.
- We can apply themes at – application level, page level or server control level.
- A theme folder has the following files :-
  1. .skin file
  2. Image folder or other resources.
- Themes are control based not html based therefor we can define and reuse any control property unlike css on html controls.
- Applying css to html is easy as compared to themes in asp because the developer needs to know which html tags will be rendered by server control.

- Applying a theme to a single page :-
  - We can instantly change the appearance of page without changing the style of server control.
  - We can apply an ASP.NET theme either downloaded or built-in on a page using the following way.

    <%@Page Theme="Red" %>

    Where Theme="Red" is theme folder name inside App_Themes.

- Applying a theme at application level :- (Global level)
  - We can apply a theme at application level by specifying the theme in web.config file.

```
<configuration>
    <System.web>
        <pages theme="Red" />
    </System.web>
</configuration>
```

- Removing or disabling theme :-
    - We can remove a theme on specific control or particular page by setting the EnableTheming to false.

- Creating our own theme :-

  We can create out theme in following way:

  1. Create a predefine folder App_Themes in the application.
  2. For every theme, that we want to create, we create a folder with a name in App_Themes.
  3. For every theme folder we require the following resource –
     i. Single skin file
     ii. A css file
     iii. Images folder

❖ Skin :-

- A Skin is a file that contains the visual properties for look & feels for ASP.NET server controls.
- It has an extension .skin.
- It contains the properties for individual controls like buttons, textboxes, labels etc.
- We can define a skin in a separate file for every control or define all skin in a single file for a theme.
- Skins are of 2 types :
  1. Default Skin
  2. Named Skin

1. Default Skin :-

- It is applied automatically to all the controls of the same type when a theme is applied on a page.
- Here the SkinID property is not defined.
- Only one default control skin for control type is allowed in the same theme.

  <asp:Textbox runat=”server” ForeColor=”red”>
  </asp:Textbox>

2. Named Skin :-

- It is a control skin with the SkinID property.
- Named Skin does not apply to control automatically.
- The SkinID should be unique because duplicate SkinID for control type are not allowed in same theme.
  <asp:Textbox runat=”server” ForeColor=”red” BackColor=”#ffffcc” BorderColor=”blue” SkinID=”stxt” />

❖ Web Configuration file :-

- Web.config file holds the information used by the application to controls its behavior.
- It simply holds keys and value pair that are recognized by asp.net.
- The values are easily modifiable and we can also add own customized key-value pairs to control other settings that are not handled implicitly by ASP.NET.
- If we want to provide any setting for the entire application, we place this file in the root application folder.
- Like global.asax, it also prevents us from being accessed via a web browser or a client.
- Changes to this file are automatically detected and application restart automatically.
- It can be thought as version of the registry settings like in windows application.
- It has no html or script blocks. It is just purely XML.
- The Web.config file contains the following sections.

  1. <config> section :-
     - This section declares the XML data contained in a file and returns an appropriate object based on the given data for processing.
     - It is denoted by <config>.

  2. <System.net> section :-
     - It contains information on network services for ASP.NET runtime.

  3. <sitemap> section :-
     - It provides the settings for sitemap file configuration.

  4. <sessionstate> section :-
     - It provides the setting for customizing the session state.
     - E.g : We can store session variable on completely separate machine so that whenever the server crashes, we can recover session data.

  5. <authentication> section :-
     - It provides the configuration setting for forms authentication.

  6. <browsercaps> section :-

- It configure sections like html, header etc also represents information about the browser.
- E.g JavaScript browser version etc.

7. <System.web> :-

- This tag considers all the setting required for Asp.net application.

8. <pages> :-

- It allows controlling the page object at the application level rather than at page level.
- We can set a theme at application level using <pages> section.

9. <webservice> :-

- It configures the setting for web services.

- Note: - Other than the above section we also have <appSettings>, <location>, <connectionString> etc for configuring database connections, applications etc.